Oven Simulation

```
Read set temperature and status of oven.
Read current temperature of oven.
If the oven status is closed, then:
        Call ON (arguments: set temperature, current temperature)
ENDIF
ON (parameters: set temperature, current temperature)
    Set highest limit = set temperature * 1.10
    Set increment = 0.001
    While current temperature < highest limit
        Increment Current temperature.
        Display the current temperature and “OVEN ON”
    ENDWHILE
    Call OFF (arguments: set temperature, current temperature)
END ON
OFF (parameters: set temperature, current temperature)
    Set lowest limit = set temperature * 0.9
    Set decrement = 0.001
    While current temperature > lowest limit
        Decrement Current temperature.
        Display the current temperature and “OVEN OFF”
    ENDWHILE
    Call ON (arguments: set temperature, current temperature)
END OFF
```

<u>All Squares</u>

```
Set counter = 0
Set center of grid = (1024,1024)
Set center of square = center of grid
Set top left corner = (0,0)
Set bottom right corner = (2048,2048)
Read size(k) from the input file.
Read the point coordinates from the input file.
If k and point = 0
    END program
Else
    Call find (arguments: k, center of square)
ENDIF
find (parameters: k, center of square)
    If point in the square whose center is the center of grid and of size k
        Increment counter.
    ENDIF
    If k <=1 or the center of new square is out of the grid
        Return counter.
    ENDIF
    Call find (arguments: k/2, top left corner of the square)
    Call find (arguments: k/2, top right corner of the square)
    Call find (arguments: k/2, bottom left corner of the square)
    Call find (arguments: k/2, bottom right corner of the square)
END find
Display counter.
```

Searching Quickly

```
Set num of words = 0
Set num of titles  = 0
While true
    Read words to ignore.
    Increment num of words.
    If words to ignore is ::
        Break.
ENDWHILE
While true
    Read titles.
    Increment num of titles.
    If titles is stop
        Break.
ENDWHILE
Call KWIC_index (arguments: titles, words to ignore, num of titles, num of words)
KWIC_index(parameters: titles, words to ignore, num of titles, num of words)
    Set counter = 0
    For h = 0 to num of titles
        Set separated words = Tokenize " \n" in titles[h]
        While separated words is not NULL
            Call lower (arguments: separated words[counter])
            Increment counter.
        ENDWHILE
        For k = 0 to counter
            Set ignore = 0
            For I = 0 to num of words
                If separated words[k] is in words to ignore
                    Ignore = 1
                ENDIF
            ENDFOR
            If not ignore
                Call upper(arguments: separated words[k])
                For j = 0 to counter
```

```
                    Display words separated[j]
                ENDFOR
                Call lower (arguments: separated words[k])
            ENDIF
        ENDFOR
        Counter = 0
    ENDFOR
END KWIC_index
lower (parameters: separated word)
    lowercase all letters of the word
END lower
upper (parameters: separated word)
    uppercase all letters of the word
END upper
```

Treasure Everywhere

```
Set map num = 0, count = 0
While True
    Read steps from the input file.
    If steps is END
        Break.
    ENDIF
    If last letter of steps is "."
        Increment map num
        Call Directions (arguments: steps, map num, count)
    Else
        Increment count
    ENDIF
ENDWHILE
Directions (parameters: steps, map num, count)
    Set x = 0, y = 0, num
    For I = 0 to count
        num = number in steps
        Call RemoveDigits (arguments: steps[i])
        Toknize steps to remove "." and ","
        If steps is "N"
            Increment y by num.
        Else if steps is "S"
            Decrement y by num.
        Else if steps is "W"
            Decrement x by num.
        Else if steps is "E"
            Increment x by num.
        Else if steps is "NE"
            Increment x by num * cos (PI / 4)
            Increment y by num * cos (PI / 4)
        Else if steps is "NW"
            Decrement x by num * cos (PI / 4)
            Increment y by num * cos (PI / 4)
        Else if steps is "SE"
            Increment x by num * cos (PI / 4)
            Decrement y by num * cos (PI / 4)
        Else if steps is “SW”
            Decrement y by num * cos (PI / 4)
            Decrement x by num * cos (PI / 4)
        ENDIF
    ENDFOR
    Call calculate (arguments: x, y, map num)
END Directions
calculate (parameters: x, y, map num)
Set distance = sqrt (x^2 + y^2)
Display map num , x, y and distance.
END calculate
```